MODEL CHECKING PADA PROTOKOL BERMAN DAN GARAY

Sheila Nurul Huda, S.Kom
[1]*Jurusan Teknik Informatika, Fakultas Teknologi Industri,Universitas Islam Indonesia*
Jl. Kaliurang Km. 14 Yogyakarta 55501
Telp. (0274) 895287 ext. 122, Faks. (0274) 895007ext. 148
E-mail: sheila@staff.uii.ac.id

ABSTRAK
Distributed consensus merupakan permasalahan yang dikenal dalam sistem terdistribusi dimana terdapat beberapa proses yang berjalan secara konkuren, tiap proses memulai dengan nilai awalnya, dan pada akhir proses mereka harus mencapai nilai yang sama. Artinya, semua proses harus mencapai kesepakatan, sekalipun nilai awal mereka mungkin berbeda. Terdapat kondisi yang memungkinkan terjadinya kegagalan pada proses yang terlibat, kondisi ini disebut byzantine failure. Protokol Berman dan Garay dirancang untuk menyelesaikan distributed consensus dengan kehadiran byzantine failure. Untuk mendapatkan kepastian secara formal bahwa protokol tersebut mencapai distributed consensus dilakukan verifikasi dengan metode model checking menggunakan SPIN model checker.

Kata Kunci: distributed consensus, byzantine failure, model checking, protokol Berman dan Garay

1. LATAR BELAKANG

1.1 Distributed Consensus dengan Byzantine Failure

Pada banyak komputasi terdistribusi terdapat kebutuhan bagi entitasnya untuk membuat keputusan yang lokal namun terkoordinasi. Keputusan yang terkoordinasi ini disebut sebagai agreement. Pada permasalahan *p-Agreement* (**Agree(p)**), setiap entitas x memiliki nilai input v(x) dari suatu himpunan nilai tertentu (biasanya {0, 1}), dan harus pada akhirnya menentukan suatu nilai d(x) dari himpunan tersebut dalam waktu yang terbatas. Permasalahan *p-Agreement* adalah untuk memastikan bahwa terdapat sedikitnya p buah entitas yang membuat keputusan yang sama. Terdapat pula syarat tambahan (yang disebut *nontriviality* atau *validity*) bagi permasalahan ini, yaitu apabila semua nilai awal adalah sama, maka keputusan yang dibuat adalah sesuai nilai tersebut (Santoro, 2007).

Berdasarkan nilai p, terdapat beberapa tipe agreement. Misalnya jika p lebih dari (n/2) + 1, disebut sebagai *strong majority*. Jika p = n, permasalahan ini disebut sebagai konsensus. Permalasahan konsensus terjadi dalam berbagai aplikasi. Misalnya pada pesawat terbang dimana beberapa sensor digunakan untuk menentukan apakah sudah saatnya untuk menjatuhkan kargo sekarang. Dimungkinkan beberapa sensor mendeteksi "ya", sementara ada sensor yang mendeteksi "belum saatnya". Berdasarkan nilai inilah suatu keputusan harus dibuat untuk menentukan apakah kargo harus dijatuhkan sekarang atau tidak. Strategi solusi untuk contoh ini adalah menjatuhkan kargo jika semua sensor mendeteksi "ya", atau menjatuhkan kargo jika setidaknya satu sensor mendeteksi "ya". Solusi pertama mengkomputasikan seluruh nilai sensor dengan operator AND, sementara solusi kedua menggunakan operator OR. Jika nilai awal sama, maka hasil akhir menggunakan kedua solusi ini sama dengan nilai awalnya. Tampak jelas bahwa konsensus dapat dicapai dengan mudah (misalnya dengan menghitung AND atau OR dari nilai tersebut).

Penelitian ini bertujuan untuk mempelajari dan memodelkan permasalahan pencapaian konsensus pada sistem terdistribusi. Pada permasalahan konsensus, tiap proses memulai dengan nilai awalnya, dan pada akhir proses mereka harus mencapai nilai yang sama. Artinya, semua proses harus mencapai kesepakatan, sekalipun nilai awal mereka mungkin berbeda.

Hanya saja terdapat kemungkinan terjadinya kegagalan yang menyebabkan konsensus sulit tercapai. Kegagalan dalam pencapaian konsensus diklasifikasikan menjadi beberapa macam, yaitu link failure, stopping failure, serta byzantine failure. Link failure adalah jika jaringan yang menjadi penghubung sistem terdistribusi mengalami kegagalan sehingga pesan yang dibawa hilang dalam jaringan. Stopping failure jika terdapat proses yang rusak dan berhenti bekerja. Sementara byzantine failure adalah jika terdapat proses yang mengalami kegagalan, yang menyebabkannya berperilaku tidak sesuai dengan yang seharusnya. Kata byzantine sendiri pertama kali digunakan dalam paper Lamport, Pease, and Shostak, dimana permasalahan konsensus diformulasikan dalam skenario jenderal byzantine. Beberapa jenderal bizantium yang berada pada tempat terpisah menghadapi musuh yang sama dan harus menyepakati apakah mereka akan menyerang atau tidak. Para jenderal mengirimkan pesan mengenai pendapat mereka satu sama lain menggunakan prajurit yang berjalan kaki membawa surat. Pada link failure, kegagalan mungkin terjadi jika prajurit pembawa pesan ditangkap musuh dan dibunuh,

sehingga pesannya tidak sampai. Pada stopping failure, salah satu atau lebih jenderal terbunuh karena diserang musuh terlebih dahulu. Sementara dalam byzantine failure yang menjadi masalah bukanlah channel pengiriman yang tidak reliable, melainkan perilaku jenderal (sebagai model dari proses) yang mungkin mengirim pesan yang berbeda pada jenderal yang berbeda. Di satu waktu ia berkata serang, di lain waktu ia berkata jangan (Lynch, 1996).

Terdapat beberapa algoritma yang telah ditawarkan untuk mencapai konsensus dalam keadaan terdapat byzantine failure. Diantaranya adalah protokol yang dirancang oleh Berman dan Garay. Ketercapaian konsensus pada protokol ini sulit dibuktikan secara intuitif, terutama jika jumlah proses cukup besar. Penelitian ini akan membahas pembuktian secara formal terhadap ketercapaian konsensus melalui protokol Berman dan Garay.

1.2 Protokol Berman dan Garay

Berman dan Garay (1989) merancang suatu protokol yang bersifat fault tolerant, yang memungkinkan proses-proses tersebut mencapai konsensus. Terdapat N proses yang bersifat reliable dan K proses yang bersifat unreliable, dimana N haruslah lebih besar dari 3*K dan K paling sedikit 0. Tidak ada suatu cara untuk bisa membedakan proses mana yang merupakan proses yang unreliable. Semua proses berkomunikasi melalui pengiriman pesan. Setiap proses memiliki variabel lokal, yang nilai awalnya 0 atau 1.

Proses-proses tersebut berkomunikasi satu sama lain dalam ronde. Tiap ronde terdiri dari dua fase pengiriman pesan: pada ronde i, i > 1, pada fase pertama setiap proses mengirimkan nilai variable lokalnya ke semua proses, termasuk dirinya; pada fase kedua, proses i mengirimkan nilai mayoritas yang diterimanya pada fase pertama (agar nilai mayoritas dapat didefinisikan dengan baik, maka nilai N+K adalah ganjil) ke semua proses. Jika suatu proses menerima N, atau lebih, instance dengan nilai yang sama pada fase pertama pada suatu ronde, maka proses tersebut mengubah nilai variabel lokalnya menjadi nilai ini; selain dari itu maka ia akan mengubah nilai variabel lokalnya dengan nilai yang diterima (dari proses i) pada fase kedua ronde ini.

Tujuan dari protokol ini adalah pada akhir ronde K+1 telah mencapai syarat ketercapaian konsensus, yaitu:

a. Pada akhirnya semua reliable proses memiliki nilai yang sama pada lokal variablenya.
b. Jika semua proses yang reliable memiliki nilai awal (lokal variabel) maka nilai akhir mereka sama dengan nilai awal mereka bersama.

1.3 Model Checking dengan SPIN

Untuk menjamin reliabilitas suatu sistem, diperlukan perancangan yang hati-hati, terutama bagi sistem yang bersifat kritis. Reliabilitas perangkat lunak adalah kemungkinan operasi program komputer bebas kegagalan di dalam suatu lingkungan tertentu dan waktu tertentu. Sommerville (2001) menyatakan bahwa perangkat lunak yang bebas dari kesalahan adalah perangkat lunak yang dengan tepat mengikuti spesifikasinya.

Diperlukan semantik formal untuk mendefinisikan model sistem yang tidak bersifat ambigu, sehingga dapat diketahui correctness dari sistem tersebut. Correctness dapat dicapai dengan teknik verifikasi. Terdapat dua metode yang digunakan dalam verifikasi formal, yaitu model checking dan theorem proving.

Model checking merupakan teknik efektif untuk mengamati kesalahan desain yang potensial terjadi. Teknik ini menggunakan metode formal untuk memverifikasi semua finite state dalam sebuah sistem dan akan memverifikasi kebenaran dari spesifikasi sistem yang diekspresikan dalam logika temporal. Model checking bekerja secara otomatis dan relatif cepat. Selain itu, jika terdapat error dalam desain, model checking akan memproduksi sebuah counterexample yang bisa digunakan sebagai penunjuk sumber error (Baier dan Katoen, 2008).

SPIN merupakan tool model checker yang memiliki keistimewaan dalam pengecekan terhadap sistem yang konkuren/paralel. Selain memiliki input bahasa PROMELA, SPIN mampu mendefinisikan spesifikasi formal dalam bentuk assertion dan Linear Time Temporal Logic Property (LTL property). Sebagai tool model checker, SPIN mengecek semua state space yang finite dan mampu memverifikasi terhadap spesifikasi sistem yang telah didefinisikan sebelumnya (Holzmann, 2003).

Dalam penelitian ini akan dilakukan pemodelan terhadap protokol Berman dan Garay menggunakan bahasa PROMELA dan melakukan pengecekan menggunakan SPIN model checker untuk mengetahui apakah protokol tersebut telah memenuhi spesifikasinya untuk menjamin ketercapaian konsensus.

2. MODEL PROMELA

2.1 Model Sistem

Model sistem dibuat menggunakan bahasa Promela, sebagaimana tampak dalam gambar 1.

```
#define N 4
/* number of reliable processes */
#define K 1
/* number of unreliable processes */
#define M 5
/* total number of processes (i.e.,
N+K) */
/* M must be odd for majority to be
well-defined */

#define RANDOM(x) \
   if \
   :: x = 1 \
   :: x = 0 \
   fi \
```

```
typedef Achan {
   chan ch[M] = [1] of {bit};
}

Achan Mchan[M];

proctype P(byte num) {
   bit c;
   byte i,
        j,
        W;

   atomic { i = 0; RANDOM(c); }
   do
   :: else -> break;
   :: i < K + 1 ->
        atomic {
           j = 0;
           do
           :: j < M -> if
                :: num < K -> RANDOM(c);
                 :: num >= K -> skip;
                       fi;

Mchan[num].ch[j]!c; j++;
           :: j == M -> break;
           od;
        }

        atomic{ j = 0; W = 0; }
        do
        ::   j   <   M   ->   atomic{
Mchan[j].ch[num]?c; W = W + c; j++; }
        :: j == M -> break;
        od;

        if
        :: i == num ->
              atomic {
                 j = 0;
                 do
                 :: j < M -> c = (W >
(M-W));    /* majority */
                       if
                        ::  num  <  K  ->
RANDOM(c); /* select random value */
                        ::  num  >=  K  ->
skip;
                       fi;

Mchan[num].ch[j]!c; j++;
                 :: j == M -> break;
                 od;
              }
        :: else -> skip
        fi;

        atomic{
           Mchan[i].ch[num]?c;
           if
           :: W >= N -> c = 1;
           :: (M-W) >= N -> c = 0;
           :: else -> skip;
           fi;
           i++;
/* goto next round */
        }
   od
}

init {
   byte k;

   atomic {
      do
      :: k < M -> run P(k); k++;
      :: k == M -> break;
      od
   }
}
```

Gambar 1. Model Sistem

Dalam model ini, digunakan jumlah proses yang paling minimal, yaitu unreliable proses 1 buah, dan reliable proses 4 buah (N > 3*K). Unreliable proses dimodelkan dengan menggunakan aksi nondeterministik ketika mengirim pesan pada proses lain. Untuk nomor urutan proses, maka proses yang unreliable memiliki nomor proses lebih awal. Misalnya terdapat 5 buah proses dengan 1 proses unreliable dan 4 proses reliable, maka nomor untuk proses yang unreliable adalah 0, sedangkan nomor untuk proses yang reliable adalah 1, 2, 3, dan 4.

Pada bahasa PROMELA, pengiriman pesan menggunakan channel system. Dalam model tersebut, tampak bahwa protokol Berman dan Garay berjalan pada beberapa proses secara konkuren. Proses-proses tersebut masing-masing memiliki nilai awal (yang nilainya berbeda-beda, dipilih secara random). Jumlah ronde yang terjadi adalah sebanyak K+1, yaitu 2 ronde dengan K = 1. Pada fase pertama ronde 1, tiap proses saling berkirim pesan yang berisi nilai variabel lokalnya. Untuk memperjelas pembahasan, dimisalkan proses 0 (unreliable) memiliki nilai lokal 0, proses 1, 2, 3, dan 4 memiliki nilai variabel lokal 0, 0, 1, 1 berturut-turut. Pada fase pertama ronde 1, tiap proses mengirimkan nilainya masing-masing. Unreliable proses mungkin mengirim nilai yang berbeda-beda kepada proses yang berbeda. Banyaknya kemungkinan inilah yang menyebabkan state space untuk model protokol ini menjadi besar. Dimisalkan proses 0 mengirim 1 kepada dirinya, proses 1, dan proses 4, sementara kepada proses 2 dan 3, ia mengirim nilai 0.

Proses 0 menerima dua nilai 0 dan tiga nilai 1, sehingga nilai mayoritas yang diterimanya adalah 1. Proses 1 menerima dua nilai 0 dan tiga nilai 1. Proses 2 menerima tiga nilai 0 dan dua nilai 1. Proses 3 menerima tiga nilai 0 dan dua nilai 1. Proses 4 menerima dua nilai 0 dan tiga nilai 1.

Pada fase kedua ronde 1, proses 0 mengirim nilai mayoritas yang diterimanya pada proses-proses yang lain. Karena proses 0 adalah proses yang unreliable, maka dimungkinkan ia mengirim pesan yang berbeda-beda pula. Dimisalkan proses 0 mengirim nilai 0 kepada proses 0, 2, dan 4, sementara ia mengirim nilai 1 kepada proses 1 dan 3.

Pada fase pertama ronde 1, semua proses tidak menerima nilai yang jumlahnya lebih dari sama dengan N (yaitu 4). Nilai mayoritas yang mereka terima hanya berjumlah 3. Sehingga pada fase dua ronde 1, semua proses akan mengubah nilai variabel lokalnya sesuai dengan nilai yang mereka terima dari proses 0 (yang merupakan unreliable proses). Sampai di sini, nilai variabel lokal proses 0, 1, 2, 3, dan 4 adalah 0, 1, 0, 1, 0 berturut-turut.

Pada fase pertama ronde 2, setiap proses kembali mengirim nilai variabel lokalnya. Dimisalkan proses 0 mengirim nilai 0 kepada proses 1, 2, 3, sementara ia mengirim nilai 1 kepada proses 0 dan 4.

Proses 0 menerima dua nilai 0 dan tiga nilai 1. Proses 1 menerima tiga nilai 0 dan dua nilai 1, sehingga nilai mayoritas yang diterimanya adalah 0. Proses 2 menerima tiga nilai 0 dan dua nilai 1. Proses 3 menerima tiga nilai 0 dan dua nilai 1. Proses 4 menerima dua nilai 0 dan tiga nilai 1.

Pada fase kedua ronde 2, proses 1 mengirim nilai mayoritas yang diterimanya pada proses-proses yang lain, yaitu nilai 0. Pada fase pertama ronde 2, semua proses tidak menerima nilai yang jumlahnya lebih dari sama dengan N (yaitu 4). Nilai mayoritas yang mereka terima hanya berjumlah 3. Sehingga pada fase dua ronde 2, semua proses akan mengubah nilai variabel lokalnya sesuai dengan nilai yang mereka terima dari proses 1, yaitu nilai 0.

Sampai di sini nilai variabel lokal proses 0, 1, 2, 3, dan 4 adalah 0, 0, 0, 0, dan 0 berturut-turut. Pada akhir ronde 2 inilah protokol ini mencapai konsensus, yaitu nilai 0 pada contoh ini.

2.2 Rumusan Spesifikasi Sistem dalam Assertion

SPIN akan mengecek apakah model yang telah dibangun dalam bahasa PROMELA telah memenuhi spesifikasi atau tidak memenuhi. Oleh karena itu, perlu didefinisikan spesifikasi untuk ketercapaian distributed consensus. Terdapat dua hal yang ingin dicapai dari model ini, yaitu pada akhir ronde K+1 didapati:

a. Pada akhirnya semua reliable proses memiliki nilai yang sama pada lokal variablenya.
b. Jika semua proses yang reliable memiliki nilai awal (lokal variabel) maka nilai akhir mereka sama dengan nilai awal mereka bersama.

Pertama, perlu ada mekanisme untuk mengecek apakah semua proses yang reliable telah menyelesaikan ronde K+1. Pada akhir proses, ketika suatu proses telah menyelesaikan K+1 ronde, maka akan dicek, jika proses tersebut adalah proses yang reliable, maka nilai T, yang merupakan variable untuk menyimpan jumlah proses yang telah selesai (terminated) akan naik satu. Sementara itu, jika proses tersebut adalah proses yang unreliable, maka nilai T tetap. Sehingga jika nilai T sama dengan jumlah proses yang reliable, maka semua proses yang reliable telah menyelesaikan ronde K+1.

Mekanisme untuk mengetahui apakah seluruh proses telah mencapai akhir ronde K+1 terdapat pada gambar 2.

```
bit V[N];
byte T;

atomic{
      if
      :: num >= K -> T++; V[num-K] = c;
/* reliable */
      ::    num    <    K    ->    skip
/* unreliable */
      fi;
   }
```

Gambar 2. Mekanisme Pengecekan Terminasi Proses

Kemudian, perlu diketahui pula nilai akhir dari lokal variabel masing-masing proses. Pada gambar xxx, ditunjukkan oleh syntaks `V[num-K] = c;`.

Assertion untuk tujuan pertama yang akan dibangun berupa jika-maka, yaitu jika semua reliable proses telah mencapai akhir ronde K+1, maka semua nilai variable lokal proses tersebut adalah sama. Gambar 3 menunjukkan assertion yang dibangun untuk syarat pertama.

```
#define p (T == N)
#define q ((V[0] == V[1]) && (V[1] ==
V[2]) && (V[2] == V[3]))
#define impl(p,q) (!p || q)

proctype Check() {
    assert(impl(p, q))
}
```

Gambar 3. Assertion untuk spesifikasi pertama

Syarat kedua yang akan dibuat menjadi assertion adalah jika semua proses yang reliable memiliki nilai awal (lokal variabel) maka nilai akhir mereka sama dengan nilai awal mereka bersama. Diperlukan variabel yang digunakan untuk menyimpan nilai awal masing-masing proses. Gambar 4 menunjukkan assertion untuk syarat kedua.

```
bit I[N];                /* the initial
values of the reliable processes */

#define r ((I[0] == I[1]) && (I[1] ==
I[2]) && (I[2] == I[3]))
#define s ((I[0] == V[0]) && (I[1] ==
V[1]) && (I[2] == V[2]) && (I[3] ==
V[3]))
#define impl(p,q) (!p || q)

proctype Check() {
    assert(impl((q && r), s))
}
```

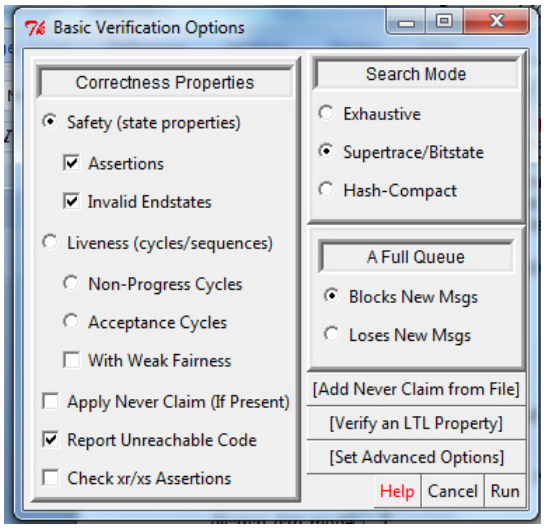
Gambar 4. Assertion untuk spesifikasi kedua

ISSN: 1907-5022

3. MODEL CHECKING DENGAN SPIN

Setelah model sistem dalam bahasa PROMELA dan spesifikasi sistem dalam assertion didefinisikan, maka pengecekan akan dilakukan secara otomatis menggunakan model checker SPIN. State space akan mengalami lonjakan yang cukup besar, karena dalam model tersebut terdapat beberapa aksi nondeterministik yang membutuhkan state lebih banyak, terutama untuk memodelkan proses yang unreliable.

Penelitian ini menggunakan XSPIN sebagai GUI dari SPIN, dan menggunakan opsi supertrace untuk mengatasi state space yang terlalu besar.



Gambar 5. Opsi Verifikasi

Hasil dari pengecekan model tersebut untuk spesifikasi pertama adalah sebagaimana ditunjukkan pada gambar 6.

```
(Spin Version 5.2.5 -- 17 April 2010)
       + Partial Order Reduction

Bit statespace search for:
       never claim                 - (not
selected)
       assertion violations        +
       cycle checks            -   (disabled
by -DSAFETY)
       invalid end states  +

State-vector  184  byte,  depth  reached
486, errors: 0
 26828638 states, stored
 22234794 states, matched
 49063432         transitions         (=
stored+matched)
 63523244 atomic steps

hash factor: 2.50139 (best if > 100.)

bits set per state: 3 (-k3)

Stats on memory usage (in Megabytes):
 4810.127    equivalent   memory   usage
for   states   (stored*(State-vector   +
overhead))
    8.000    memory used for hash array
(-w26)
    8.000    memory used for bit stack
    0.343    memory used for DFS stack
(-m10000)
   16.539    total actual memory usage

unreached in proctype Check
       (0 of 2 states)
unreached in proctype P
       line  80,  "pan_in",  state  71,
"(1)"
       (1 of 89 states)
unreached in proctype :init:
       (0 of 11 states)

pan: elapsed time 70.1 seconds
pan: rate 382468.54 states/second
pan:  avg  transition  delay  1.4297e-006
usec
```

Gambar 6. Hasil pengecekan untuk spesifikasi pertama

Hasil untuk pengecekan spesifikasi kedua adalah sebagaimana ditunjukkan pada gambar 7.

```
(Spin Version 5.2.5 -- 17 April 2010)
       + Partial Order Reduction

Bit statespace search for:
       never claim                 - (not
selected)
       assertion violations        +
       cycle checks            -   (disabled
by -DSAFETY)
       invalid end states  +

State-vector  188  byte,  depth  reached
499, errors: 0
 28744550 states, stored
 23484704 states, matched
 52229254         transitions         (=
stored+matched)
 67763029 atomic steps

hash factor: 2.33466 (best if > 100.)

bits set per state: 3 (-k3)

Stats on memory usage (in Megabytes):
 5263.284    equivalent   memory   usage
for   states   (stored*(State-vector   +
overhead))
    8.000    memory used for hash array
(-w26)
    8.000    memory used for bit stack
    0.343    memory used for DFS stack
(-m10000)
   16.539    total actual memory usage

unreached in proctype Check
       (0 of 2 states)
unreached in proctype P
       (0 of 95 states)
unreached in proctype :init:
       (0 of 11 states)
```

```
pan: elapsed time 78.2 seconds
pan: rate 367751.37 states/second
pan: avg transition delay 1.4965e-006
usec
```

Gambar 7. Hasil pengecekan untuk spesifikasi kedua

Dari gambar tersebut, tampak bahwa assertion untuk kedua spesifikasi tidak dilanggar. Pengecekan pertama 70.1 detik dengan jumlah transisi state secara keseluruhan adalah 49063432 transisi. Pengecekan kedua memakan waktu 72.8 detik dengan jumlah transisi state secara keseluruhan adalah 52229254 transisi. Model protokol Berman dan Garay telah memenuhi dua syarat ketercapaian konsensus dengan keberadaan byzantine failure dengan jumlah proses yang reliable harus lebih dari 3 kali jumlah proses yang unreliable.

Protokol Berman dan Garay mencapai konsensus dengan menggunakan fase ganda pada tiap rondenya, yaitu mekanisme untuk langsung mengubah nilai variabel lokal pada fase 1 tiap ronde jika nilai mayoritas mencapai sejumlah N (artinya terdapat nilai mayoritas pada nilai awal variabel lokal proses yang reliable), atau mengubah nilai variabel lokal pada fase 2 tiap ronde jika nilai mayoritas kurang dari N.

Konsensus dapat dicapai dalam 2 ronde (dari 5 buah proses yang terlibat) namun pesan yang dikirim mencapai (K+1)(N+1)N.

Untuk melihat pengaruh jumlah proses, maka model tersebut dicek kembali dengan kondisi jumlah proses yang reliable kurang dari 3 kali jumlah proses yang unreliable, dimisalkan jumlah proses yang reliable adalah 2 dan yang unreliable adalah 1. Gambar 8 menunjukkan hasil pengecekannya.

```
pan:1:     assertion     violated     (
!((T==2))||(V[0]==V[1])) (at depth 209)
pan: wrote pan_in.trail

(Spin Version 5.2.5 -- 17 April 2010)
Warning: Search not completed
       + Partial Order Reduction

Bit statespace search for:
       never claim               - (not
selected)
       assertion violations      +
       cycle checks        -   (disabled
by -DSAFETY)
       invalid end states  +

State-vector  88  byte,  depth  reached
224, errors: 1
    91607 states, stored
    46338 states, matched
   137945        transitions        (=
stored+matched)
   135385 atomic steps

hash factor: 732.574 (best if > 100.)
```

```
bits set per state: 3 (-k3)

Stats on memory usage (in Megabytes):
    8.037    equivalent   memory   usage
for   states   (stored*(State-vector   +
overhead))
    8.000    memory used for hash array
(-w26)
    8.000    memory used for bit stack
    0.343    memory used for DFS stack
(-m10000)
   16.539    total actual memory usage

unreached in proctype Check
       (0 of 2 states)
unreached in proctype P
       line  80,  "pan_in",  state  71,
"(1)"
       (1 of 89 states)
unreached in proctype :init:
       (0 of 11 states)

pan: elapsed time 0.175 seconds
pan: rate 523468.57 states/second
pan: avg transition delay 1.2686e-006
usec
```

Gambar 8.Hasil pengecekan untuk N < 3*K

Dari hasil tersebut tampak bahwa assertion telah dilanggar, artinya model tersebut tidak memenuhi spesifikasi yang pertama, yaitu tidak tercapainya konsensus pada proses-proses yang reliable jika jumlah proses yang reliable kurang dari 3 kali proses yang unreliable.

Berdasarkan counterexample yang disediakan oleh SPIN, assertion dilanggar karena jika nilai awal proses yang reliable berbeda, misalnya proses 1 dan 2 bernilai 0 dan 1 berturut-turut, nilai mayoritas yang diterima tiap proses pada fase 1 adalah sejumlah N (yaitu 2) dengan nilai yang bergantung pada isi pesan proses yang unreliable. Misalnya proses 0 memiliki nilai awal 0, tetapi ia mengirim nilai 0 kepada proses 1 dan nilai 1 kepada proses 0 dan 2. Maka proses 0, 1, dan 2 sekarang memiliki nilai 1, 0, dan 1 berturut-turut. Pada ronde 2, tiap proses kembali mengirim nilai variabel lokalnya. Dimisalkan proses 0 mengirim 0 kepada proses 0, dan 1 sedangkan ia mengirim nilai 1 kepada proses 2. Maka nilai variabel lokal proses 0, 1, dan 2 adalah 0, 0, dan 1. Pada akhir ronde 2, konsensus tidak tercapai.

4. KESIMPULAN

Dari pengecekan menggunakan SPIN model checker, didapati bahwa assertion tersebut tidak dilanggar. Artinya model yang telah dibangun telah memenuhi spesifikasinya. Sehingga dapat diyakini secara formal, bahwa protokol Berman dan Garay mampu menyelesaikan distributed consensus dengan byzantine failure, dengan syarat jumlah proses yang reliable lebih dari 3 kali jumlah proses yang unreliable.

PUSTAKA

Amjad, H. 2004. C*ombining Model Checking and Theorem Proving*. Technical Report UCAM-CL-TR-601, University of Cambridge, Computer Laboratory. URL http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-601.pdf.

Baier, C., Katoen, J.P. (2008). *Principles of Model-Checking*. MIT Press.

Lynch, N.A. (1996). *Distributed Algorithms*. San Fransisco: Morgan Kaufmann Publisher.

Berman, P., Garay, J.A. (1989). Asymtotically Optimal Distributed Concensus (extended abstract). In G. Ausiello, M. Dezani-Ciancaglini, dan S. Ronchi Della Rocca, editors. *Automata, Languages and Programming, volume 372 of Lecture Notes in Computer Science*, pages 80-94.

Holzmann, G. J. (2003). *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley.

Santoro, N. (2007). *Design and Analysis of Distributed Algorithm*. New Jersey: John Willey & Sons, Inc.

Sommerville, I. (2001). *Software Engineering*. Edisi 6. Erlangga, Yogyakarta.